САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

Сафиулин Искандер Радионович

Выпускная квалификационная работа бакалавра

Индексирование видеопотока для поиска схожих кадров

Направление 010300

Фундаментальная информатика и информационные технологии

Научный руководитель,

ст. преподаватель

Севрюков С. Ю.

Санкт-Петербург

2016

Содержание

Введение

В современном мире информационные технологии играют всё большую роль. Их область распространения затрагивает новые сферы человеческой жизни, что позволяет ставить и решать новые задачи. Если раньше компьютерные вычисления использовались только для решения математических вычислений, то сейчас процессоры используют даже для того, чтобы выбрать себе обед.

Не отстает в развитии и сфера распознавания изображений. Широкое распространение гаджетов позволяет людям снимать огромное количество событий с самых разных ракурсов, увеличивая базу изображений с невиданной скоростью. Благодаря столь стремительному росту появляются новые задачи, решение которых может внести очередной вклад в упрощение жизни человека. Происходит переход от простых задач сравнения изображений к более сложным задачам распознавания каких-то конкретных образов и обработки видеопотоков.

Особый интерес представляет область обработки видеопотоков. Видеофайлы имеют некую двойственную структуру. С одной стороны, это лишь последовательность изображений, и для решения многих задач можно использовать алгоритмы, которые применяются к обычным изображениям. В то же время, видеопотоки имеют некоторые отличительные особенности, например, связность, которые позволяют решать задачи с помощью эффективных алгоритмов, основывающихся на этих особенностях. Актуальность задачи обработки видеопотока и перечисленные выше особенности несут в себе необходимость более внимательного изучения способов его обработки. В данной работе рассматриваются методы индексирования видеофайлов и их применение в практических задачах работы с видеопотоками: поиске схожих кадров и синхронизации.

Чтобы рассматривать методы индексирования изображений, стоит ввести определения индексации и индекса. Индексирование в данной сфере подразумевает под собой описательную характеристику изображения – индекс, которую можно использовать потом для быстрого поиска картинки. Выделяют две больших группы методов индексирования: текстовое описание и визуальное содержание. Текстовое описание подразумевает в виде индекса набор строк, которые наиболее точно описывают изображение, например набор тэгов. Однако такой подход не всегда дает эффективный результат, потому что даже при ручном составлении индекса не все изображения поддаются словесному описанию, а автоматически это делать намного тяжелее. Поэтому рассматривать стоит вторую категорию методов индексирования – по визуальному содержанию.

Главный принцип использования индексирования видепотоков заключается в том, что результат этого процесса может многократно использоваться после окончания программы. В результате обработки видеопотока каждый кадр получает свой индекс, который можно сохранить в некотором файле, и при запуске новой задачи обработки видеопотока, например, поиске определенного кадра, использовать имеющиеся индексы для быстрого решения задачи.

В данной работе рассматриваются и анализируются принципы индексирования видеопотоков, а также тестируется их применение для задач поиска схожих кадров и синхронизации. С практическим результатом в виде кода можно ознакомиться по ссылке: https://github.com/ksander14/VKR_Safiulin

Постановка задачи

Целью данной работы является исследование различных методов индексирования изображений для поиска схожих кадров в видеопотоке, а также применение этих методов в практической задаче синхронизации видеопотоков.

Для достижения данной цели в работе будут решены следующие **задачи**:

1. Изучение предметной области.
2. Реализация алгоритмов индексирования изображения для поиска схожих кадров.
3. Практическое тестирование рассмотренных алгоритмов.
4. Поиск, разработка и тестирование алгоритмов синхронизации видеопотоков.

Рассмотренные в работе алгоритмы могут быть использованы для самых различных задач обработки видеопотоков: методы поиска схожих кадров могут быть использованы для поиска определенного объекта в множестве видеопотоков, а синхронизация видеопотоков позволяет эффективно работать с двумя видеопотоками.

Обзор литературы

В книгах [1], [2] и [3] изложены основные принципы обработки изображений: представлена общая информация о предметной области, вводятся основные понятия, изложены базовые алгоритмы обработки изображений и рассматривается решение различных задач распознавания, сравнения, трансформации изображений и др.

Работы [4] и [5] позволяют ознакомиться с принципами цифровой обработки видеопотоков, рассмотреть решения различных задач работы с видеопотоками и убедиться в актуальности темы работы.

По [6] и [7] представлено руководство по использованию библиотеки OpenCV 2.4.9, из которого можно узнать о принципах работы библиотеки и изучить стандартные функции.

В главе 1 использованы источники [8, 9, 10, 11, 12, 13, 14, 15], на основе которых будут представлены детальные разборы алгоритмов индексирования изображений, а в главе 2 разрабатываются алгоритмы с учетом наработок [17], [18].

Глава 1. Исследование алгоритмов индексации изображений

В данной главе приводятся итоги исследования предметной области и анализ существующих алгоритмов индексации изображений, а также производится практическое тестирование алгоритмов для задачи обработки видеопотоков, а именно – для задачи поиска схожих кадров в двух видеопотока.

Как было сказано во введении, первоочередный интерес представляют методы индексирования изображений по визуальному содержанию. На данный момент разработано много методов, и чтобы выбрать, какие из них нужно использовать для решения поставленной задачи, нужно проанализировать особенности видеопотока.

Во-первых, видеопоток – это последовательность изображений, и если таких видеопотоков много и они длинные, то общее количество изображений, которое требуется проиндексировать, становится огромным. Во-вторых, многие кадры из видеопотоков несут в себе и могут быть просто некачественными. Из вышеперечисленного следует, что в первую очередь стоит обратить внимание на методы, которые работают быстро и используют простую информацию, которую можно извлечь из изображения даже плохого качества.

Из, методов, которые удовлетворяют данным условиям, были выбраны три: метод цветовых гистограмм, методы матрицы переходов и метод перцептивного хеша. В следующих параграфах производится обзор этих методов.

1. 1. Метод цветовых гистограмм

Первым среди методов индексирования изображений рассматривается алгоритм построения цветовых гистограмм. Принцип действия такого алгоритма основан на подсчете попаданий пикселей в определенные диапазоны цветов, что позволяет проводить сравнение изображений путем нахождения расстояний между найденными гистограммами. Далее будет дано краткое описание алгоритма на основе материалов [8], [9], [10].

1. 1. 1. Разбиение RGB-цветов по прямоугольным параллелепипедам

Цветовое RGB-пространство рассматривается как трехмерный куб, каждая ось которого соответствует одному из трех основных цветов (красному, зеленому или синему), деления на осях пронумерованы от 0 до 255 (большее значение соответствует большей интенсивности цвета). При таком рассмотрении любой цвет RGB-изображения может быть представлен точкой куба. Для построения цветовой гистограммы каждая сторона делится на n $(n = 4)$ равных интервалов, соответственно RGB-куб делится на N $(N = 64)$ прямоугольных параллелепипедов. V_i – множество цветов, все компоненты которых попадают в определенные интервалы. Гистограмма изображения отражает распределение точек RGB-пространства, соответствующих цветам пикселей изображения, по параллелепипедам.

Выбор размерности гистограммы определяется из следующих соображений. При $n = 2$ $(N = 8)$ считались бы одинаковыми, например, $\{126,128,126\}$ и $\{0, 255, 0\}$, что, естественно, недопустимо. Установка $n = 8$ $(N = 512)$ приводит к тому, что базовая палитра становится более строгой, чем 8-битная. Такая точность не только автоматически дает некорректную обработку 256-цветных изображений, но и на остальных изображениях приводит к неестественным результатам. Очевидно, что

при росте n ситуация только ухудшается. Поэтому было установлено $n = 4$.

Оценка временной сложности

Пусть имеется изображение размера $N \times N$ пикселей. Для работы алгоритма требуется обработать каждый канал пикселя и проинкрементировать соответствующее значение вектора. Соответственно, для обработки всего пикселя компьютеру нужно произвести 4 арифметических операций, что в итоге дает в сумме $4N^2$ операций и оценку временной сложности $\Theta(N^2)$.

1. 1. 2. Виды метрик схожести гистограмм

Для определения степени схожести гистограмм существуют несколько методов [11]. В стандартном случае после нахождения гистограммы производится её нормализация: значение каждой ячейки гистограммы делится на сумму всех ячеек, что не что иное, как количество пикселей изображения.

Здесь рассматривается три формулы вычисления гистограмм. Стоит отметить, что наиболее простой является первая, однако при распознавании изображений может проигрывать другим метрикам в качестве.

1) *Пересечение гистограмм*.

Расстояние между гистограммами рассчитывается как:

$$d(H_1, H_2) = \sum_i \min\left(H_1(i), H_2(i)\right); \qquad (1)$$

Возвращаемое значение лежит в диапазоне $[0,1]$. Чем больше значение, тем более схожи изображения.

Преимуществом этой метрики считается её простота и скорость, так как для каждой пары элементов производится только одна операция сравнения, поэтому подсчет расстояния происходит достаточно быстро. Благодаря этому преимуществу этот метод получил широкое распространение.

2) *Метрика хи-квадрат*.

Формула расчета расстояния между гистограммами принимает следующий вид:

$$d(H_1, H_2) = \sum_i \frac{\left(H_1(i) - H_2(i)\right)^2}{H_1(i)}; \qquad (2)$$

Функция возвращает значение из диапазона $[0; \infty)$. Меньшее значение функции соответствует большей схожести изображений.

3) *Расстояние Бхатачария*.

Формула вычисления расстояние выглядит так:

$$d(H_1, H_2) = \sqrt{1 - \sum_i \frac{\sqrt{H_1(i) * H_2(i)}}{\sqrt{\sum_i H_1(i) * \sum_i H_2(i)}}}; \qquad (3)$$

Функция принимает значение из отрезка $[0; 1]$. Чем меньше получившееся значение, тем более схожи изображения. 0 соответствует полному сходству, 1 – наиболее сильному различию.

1. 1. 3. Модификация с использованием квадродеревьев

В алгоритмах сравнения, основанные на гистограммах цветов, можно улучшить качество поиска благодаря использованию информации о расположении цвета [9]. Это можно достичь, разбивая картинку на несколько областей и считая для каждой из них гистограмму. Тогда сравнение изображений между собой будет включать в себя попарное сравнение гистограмм соответствующих областей. Расстояние между картинками может определяться, как евклидово расстояние между векторами гистограмм: квадратный корень из суммы квадратов расстояний между гистограммами соответствующих областей. Таким образом, если для стандартного метода два изображения, отличающиеся только по взаимному расположению цветов на картинке, давали одинаковый результат, то теперь они будут различными.

Оценка временной сложности.

Пусть изображение разбивается на K областей. При составлении векторов гистограмм, так же как и в стандартном случае, нужно обойти каждый пиксель и инкрементировать одно из полей. Таким образом, составление гистограмм требует всё те же операций и оценивается как (N^2).

Если каждая гистограмма является -мерным вектором, то сравнение двух гистограмм потребует L арифметических операций, а для всего изображения KL. Получим вектор $V = (q_1, q_2 \dots q_k)$, элементы которого представляют собой разности гистограмм соответствующих областей. Наконец, считаем евклидову длину этого вектора, для чего потребуется $2K + 1$ арифметических операций. В итоге получается $K(L + 2) + 1$ операций, что дает временную сложность алгоритма (KL). Получается, что данная модификация дает ухудшение в скорости поиска изображений в K раз по сравнению с обычной версией алгоритма. Поэтому в случае применения данной модификации проводятся тестирования с различными значениями K для выявления оптимального, при котором обеспечивается требуемое качество без критической потери в скорости.

1. 2. Матрицы переходов как текстурный признак сравнения

Помимо методов, основывающихся на цветовом сравнении изображений, есть и иные идейные подходы поиска схожих изображений. В частности, характеристикой изображения может служить её текстура.

Матрицы переходов – один из способов описания текстуры изображений, который позволяет представить текстуру изображения в

виде многомерного вектора [12]. Алгоритм построения матрицы переходов описывается следующими шагами.

Шаг 1. Пусть имеется изображение размером $N \times M$. Сначала строится матрица той же размерности, которая задает уровень яркости каждого пикселя.

Шаг 2. Происходит квантование яркости на несколько уровней. Например, значения яркости для пикселя равно 245. Яркость квантуется на 4 области, получается 4 новых уровня яркости с индексами 0, 1, 2 и 3. Значение 245 попадает в область 192-255, поэтому в новой матрице для данного пикселя запишется число 3.

Шаг 3. Вычисляются матрицы переходов. Их размерности равны $k \times k$, где k- уровни квантования. Для предыдущего примера $k = 4$. У каждой матрицы смежности есть направление. Например, если направление матрицы равно $(x, y) = (1,0)$, то она характеризует переходы пикселей вдоль оси x, то есть направо. Значение i −ой строки и j −ого столбца матрицы перехода равно количеству переходов значения уровня яркости $(i - 1)$ в значение уровня яркости $(j - 1)$ по направлению (x, y). Например, значение элемента первой строки первого столбца матрицы переходов по направлению $(1, 0)$ показывает, сколько пикселей в квантованной матрице со значением 0 имеют справа пиксель с тем же значением 0.

Матрицы переходов вычисляются по нескольким направлениям. В стандартном случае вычисляют 4 матрицы: по направлениям вправо, влево, вверх, вниз.

Шаг 4. Полученные матрицы преобразуются в вектор размерности 128.

Наглядным образом работу алгоритма на примере демонстрирует рисунок ниже (рис. 1).

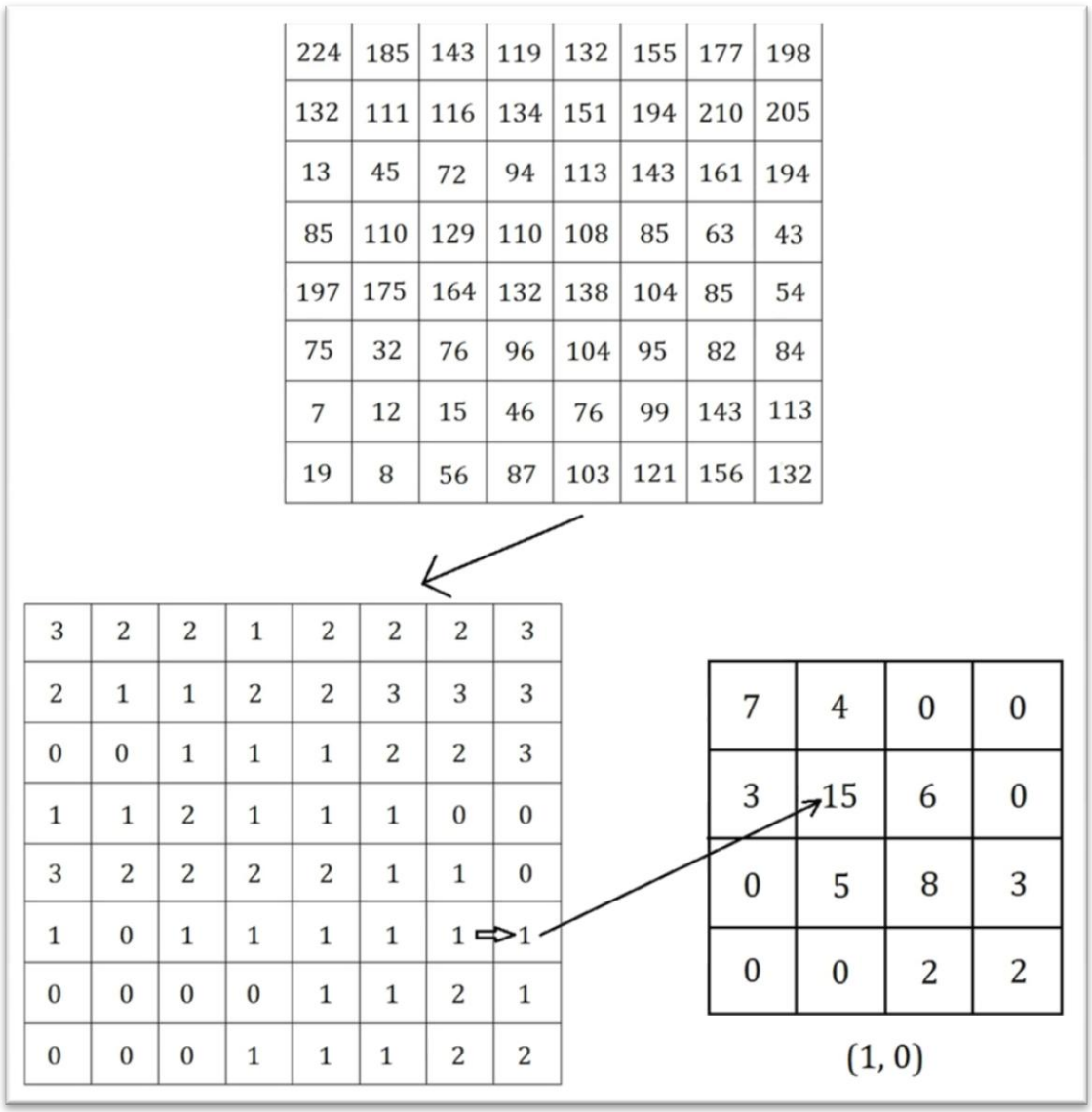

Рис. 1. Построение матрицы переходов

Чтобы произвести сравнение изображений с помощью матриц переходов, находят расстояние между векторами с помощью какой-либо метрики, чаще всего L_1. Чем меньше это значение, тем более похожи изображения по своей структуре.

Оценка временной сложности

Рассматривается этап построения вектора признаков. Пусть изображение имеет размер $N \times N$ пикселей. Сначала требуется перевести изображение в градации серого, что потребует $3N^2$ операций. Для получения квантованной матрицы яркости потребуется обработать каждый пиксель и произвести одну арифметическую операцию, что дает

N^2 операций. После этого строится 4 матрицы переходов, для которых должны обработаться все элементы квантованной матрицы 4 раза, при каждой обработке инкрементируя соответствующий элемент матрицы переходов. Получается $4N(N-1)$ операций инкремента.

Суммируя полученные данные, получаем порядка $8N^2$ операций и оценку сложности алгоритма $\theta(N^2)$. Она аналогична оценке сложности метода цветовых гистограмм, однако коэффициент при старшем члене в данном случае выше, поэтому скорость алгоритма должна быть ниже.

1. 3. Сравнение изображений с помощью перцептивных хешей

Алгоритм перцептивного хеша используется для разных задач обработки изображений. Перцептивный хеш – это такая функция, генерирующая некоторый отпечаток изображения, который удобно сравнивать с отпечатками других изображений. Существует много вариантов реализации этой функции, автором, на основе изученных материалов [12], [13], [14], рассмотрен один вариант функции, оптимальный с точки зрения скорости, описание которого представлено ниже.

Алгоритм построения простого перцептивного хеша (Simple Hash) изображения состоит из следующих шагов:

1) уменьшение размера исходного изображения до $K \times K$ пикселей;
2) перевод уменьшенного изображения в градации серого;
3) вычисление среднего значения цвета всех пикселей;
4) бинаризация всех пикселей изображения по найденному среднему значению: если значение пикселя больше среднего, то новое значение равно 1, иначе – 0;
5) перевод массива битов в цепочку длины K^2.

Для сравнения полученных хешей используется расстояние Хэмминга: оно равно количеству битов различающихся битов в сравниваемых цепочках. Чем меньше это значение, тем более похожи изображения.

Значение параметра K может быть разное. Из имеющихся исследований [14] можно сделать вывод, что обычно используются значение в диапазоне от 8 до 32. Желательно, чтобы значение было степенью 2: при уменьшении изображения будет удобнее его разбивать на блоки интерполирования.

Оценка временной сложности

Пусть исходное изображение имеет размер $N \times N$ пикселей. Первым шагом производится уменьшение размера изображения. Несмотря на разнообразие вариантов осуществления данной операции, для этого в любом случае потребуется N^2 арифметических операций. Для перевода изображения в градации серого требуется $3K^2$ арифметических операций, подсчета среднего значения – K^2, бинаризации – еще $2K^2$. В сумме получаем $(N^2 + 6K^2)$ операций. Учитывая, что $K \ll N$ в большинстве случаев, в итоге получается оценка сложности $\theta(N^2)$, причем коэффициент старшим членом получается меньше, чем в рассмотренных ранее методах, поэтому с теоретической точки зрения этот алгоритм должен работать быстрее всех.

1. 4. Реализация и тестирование алгоритмов

Тестирование рассмотренных методов индексирования изображений производится на задаче поиска схожего кадра в видеопотоке по образцу. Будет использоваться набор пар видефайлов. В каждой паре видеофайлов фигурируют одни объекты. Задача состоит в том, чтобы по выбранному кадру из одного видеопотока найти кадр из другого, наиболее похожего на данный.

Тестовые испытания были проведены на наборе пар любительских видеозаписей. Конфигурация компьютера представлена ниже:

Процессор: Intel Core i3-2350M 2.30Ghz x64

Память: 4Gb DDR3

ОС: Windows 10 x64

Код реализации алгоритмов компилировался в Visual Studio 2015 с использованием библиотеки OpenCV 2.4.9 x64 для операционной системы Windows.

Далее описывается, какие функции библиотеки OpenCV использовались для тестов.

Сначала рассматривается тестирование метода на основе цветовых гистограмм. Для вычисления гистограммы каждого изображения используется функция из библиотеки OpenCV *calcHist*, которая считает гистограмму для данного изображения. В качестве аргументов функции можно указать, по каким диапазонам каждого канала нужно вести подсчет и на сколько областей делить каждый канал. Для тестов используется весь диапазон для каждого из трех цветов - [0, 255], деление каждого канала производится на 4 части. В итоге получается матрица $4 \times 4 \times 4$ (всего 64 элемента).

После использования этой функции производится нормализация полученной матрицы с помощью функции *normilize*. На этом заканчивается процесс вычисления гистограммы изображений.

Непосредственно сравнение гистограмм производится с помощью функции *compareHist*, аргументами которой являются две гистограммы, а также целое число, которое указывает, какой метод нужно использовать. Библиотекой предусмотрен выбор из всех трех методов сравнения гистограмм, которые были рассмотрены в работе.

Кроме этого, используется собственная реализация функции деления изображений. Пусть N - количество делений одного измерения

изображения. Если $N = 2$, то изображение делится на 4 прямоугольника, если $N = 4$ - на 16 т.д. Под $N = 1$ подразумевается изображение без делений. Тесты были проведены для $N = 1,2,4,8$.

Для каждого варианта производился замер времени индексирования видео и поиска наиболее похожего кадра каждым из трех методов, а также качественного результата как удачно/неудачно. По итогам всех тестирований считались средние арифметические значения по каждому из показателей.

Общие результаты тестирования приведены в таблицах 1, 2, 3. Так как тестовые замеры производились на видеозаписях разных размерностей кадров, для определения среднего арифметического были выбраны указанные величины измерения.

Таблица 1. Время индексирования (мс/кадр 1280×720))

N	Расстояние Бхатачария
1	17.8
2	18.3
4	19.2

Таблица 2. Время поиска (мс/1000 кадров)

N	Расстояние Бхатачария	Пересечение гистограмм	Метрика Хи-квадрат
1	3.8	5.6	3.6
2	15.7	22.2	14.5
4	64.5	84.8	54.7

Таблица 3. Качество поиска

N	Расстояние Бхатачария	Пересечение гистограмм	Метрика Хи-квадрат
1	70%	39%	55%
2	51%	49%	45%
4	60%	66%	73%

Для тестирования метода, основанного на матрицах перехода, потребовалось реализовать алгоритм вручную, так как в библиотеке OpenCV для C++ нет имплемента, реализующего данную функцию.

Тестирования производились по аналогичной с методом гистограмм схеме.

Таблица 4. Результаты тестов метода матриц переходов

Индексирование (мс/кадр 1280×720)	Поиск (мс/1000кадров)	Результат
154.5	2.4	60%

Наконец, ниже представлены результаты тестов алгоритма перцептивного хеша. Ввиду отсутствия реализации алгоритма в OpenCV, код, так же как и для метода матриц переходов, писался вручную.

Таблица 5. Результаты тестов перцептивного хеша

Значение K	Индексирование (мс/кадр 1280×720)	Поиск (мс/1000кадров)	Результат
8	10.6	0.7	31%
32	10.8	9.2	42%

Итоговый анализ результатов тестирования

Благодаря теоретическим исследованиям и реализации алгоритмов и их тестированию удалось получить более точное представление о рассмотренных методах индексирования изображений.

Среди метрик для сравнения гистограмм изображений выделяется расстояние Бхатачария. Несмотря на громоздкость своей формулы, в реализации библиотеки OpenCV она работает примерно так же быстро, как остальные, но качественный результат дает сравнительно выше. При этом модификация с использованием квадродеревьев является для

данной метрики скорее даже минусом, нежели чем плюсом, так как процент качества поиска при делении изображений ниже, чем без деления.

Две другие метрики демонстрируют в целом схожие результаты. Для них квадродеревья дают прирост в качестве, тем самым позволяя модифицировать количество делений изображения в поисках баланса между скоростью и качеством.

Метод матриц переходов показал по качеству результат в среднем такой же, что и методы гистограмм. Если анализировать скорость, можно заметить, что время поиска оказалось быстрее любой из метрик гистограмм, однако время индексирования на порядки больше. Подобный результат легко объясняется теоретически – из анализа временной сложности можно заметить, что хоть методы гистограмм и текстур имеют один порядок сложности, коэффициент C в оценке метода матриц переходов заметно выше аналогичного коэффициента в оценке метода гистограмм для любой из метрик.

Алгоритм перцептивного хеша показал хорошие результаты только в плане скорости, подтвердив теоретическую оценку алгоритма. В качестве же поиска алгоритм отстал от остальных, что подтверждает уже проведенные исследования о его полезности для узкого спектра задач обработки изображений [6]. И поиск схожих кадров, как показали тесты, в этот спектр задач не входит.

Если метод гистограмм и перцептивный хеш раньше тестировались и использовались для различных задач [8], [9], [10], [13], [14], то метод матриц переходов не пользовался такой популярностью, даже в плане исследований. В ходе исследования, проведенного в этой работе, удалось оценить работу этого метода и провести его сравнительную характеристику с другими методами на данном наборе тестовых данных.

Глава 2. Применение индексации для задачи синхронизации видеопотоков

В данной главе рассматривается применение алгоритмов индексации изображений для задачи синхронизации видеопотоков. В данной главе помимо применения самой индексации рассматриваются разные подходы решения задачи синхронизации, основанные на характерных особенностях видеофайлов и полученные после изучения существующих наработок [9], [17], [18].

В современном мире часто самые различные системы используют записи с видеокамер, например для охраны зданий или отслеживания посетителей. Обычно такие видеокамеры имеют синхронизацию на аппаратном уровне: получаемые видеофайлы имеют одинаковое стартовое время и частоту кадров, и при их просмотре демонстрируется одно и то же действие.

Однако подобная аппаратная синхронизация не всегда имеет место. Может получиться так, что имеются записи с видеокамер, которые созданные в разное начальное время или имеющие различные частоты кадров в разное начальное время. А для того, чтобы эффективно работать с такими файлами, видеопотоки должны быть синхронизированы. В такой ситуации возникает задача синхронизации видеопотоков вручную или автоматически.

В этой главе будет рассмотрена задача автоматической синхронизации видеопотоков.

2. 1. Постановка задачи

Задачу синхронизации видепотоков для случая, описанного во введении, можно сформулировать следующим образом.

Даны два видеофайла. Каждый из них представляет собой записи статичных видеокамер, которые имеют некоторое пересечение по области и по времени съемки: минимум 80% процентов по области съемки и минимум 50% по времени более короткого видео. При этом видеопотоки не синхронизированы – при одновременном просмотре видеофайлов не будет наблюдаться одно и то же событие.

Предполагается, что каждый видеофайл имеет постоянную частоту кадров. Тогда временное несоответствие между видеопотоками происходит, если они имеют разные частоты кадров или из-за сдвига между их начальными моментами. Значит, существует линейная зависимость между временными линиями, которую математически можно выразить следующей формулой:

$$T_1 = \alpha T_2 + \Delta\,; \qquad (5)$$

где T_1 – время первого видеопотока, T_2 – второго, α – отношение частоты кадров первого видео к частоте кадров второго, Δ – временной сдвиг начальных моментов.

Тогда задачу синхронизации видеопотоков заключается в нахождении α и Δ. Если ввести предположение, что видеофайлы имеют одинаковые частоты, то задача сводится к нахождению Δ.

2. 2. Варианты реализации

Для решения поставленной задачи были разработаны и проанализированы два принципиальных подхода к решению задачи.

Первый подход к решению заключается в прямом сравнении изображений двух видеопотоков, используя методы индексирования, рассмотренные в первой главе.

Второй же подход основывается на связности видеопотоков и решает задачу путем сравнения последовательно идущих кадров внутри каждого файла и последующего сравнения полученных результатов между видеопотоками.

В следующих параграфах представлены реализации этих идей для разных вариантов методов индексации изображений.

2. 2. 1.Алгоритм прямого сравнения кадров видеопотоков

При анализе постановки задачи в первую очередь приходит идея прямого сравнения изображений между видеопотоками. На основе этих сравнений можно построить модель сравнения последовательностей кадров любой длины, чтобы при конкретной длине найти последовательности, которые являются наиболее схожими среди всех выбранных, что позволит найти временной сдвиг.

Подобный подход можно описать следующей последовательностью шагов:

1. Проиндексировать все кадры обоих видеопотоков.
2. Сравнить последовательности кадров видеопотоков с помощью имеющихся индексов и найти наиболее похожие.
3. Приняв, что наиболее похожие подпоследовательности кадров являются совпадением видеопотоков, найти временной сдвиг.

На математическом языке алгоритм можно описать следующим образом. Пусть есть некоторая метрика, с помощью которой производится сравнение изображений. Пусть $M -$ количество кадров первого видеофайла, $N -$ второго. Не умаляя общности, предположим, что $M < N$.

Пусть $L -$ количество кадров, по которому у видеопотоков происходит пересечение по времени (рис. 2).

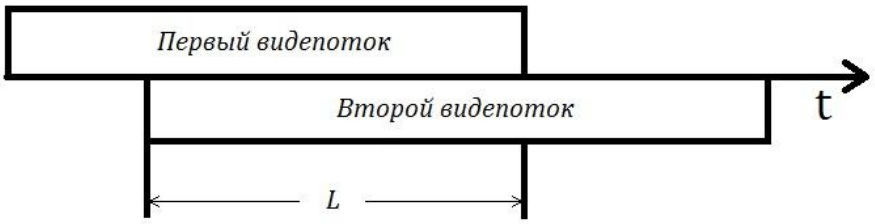

Рис. 2. Пересечение кадров видеопотоков

В силу начальных условий задачи, пересечение происходит минимум по 50% времени более короткого видеопотока, то есть $\min(L) = \lceil 0.5M \rceil$, а $\max(L) = M$, так как в лучшем случае первое видео полностью попадет по времени во второе.

Чтобы найти оптимальный временной сдвиг, требуется произвести перебор всевозможных временных сдвигов, который описывается следующим итеративным подходом:

1) стартовая позиция алгоритма: временной сдвиг второго видео составляет ($\lfloor 0.5M \rfloor$) (рис. 3):

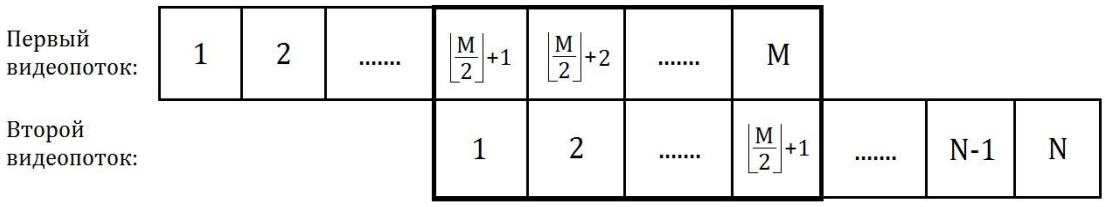

Рис. 3. Первая итерации алгоритма

2) в каждой последующей итерации первый кадр первого видеопотока смещается по временной шкале на одну позицию вправо относительно второго видеопотока. Таким образом, шкала пересечения подпоследовательностей кадров для второй итерации выглядит так (рис. 4):

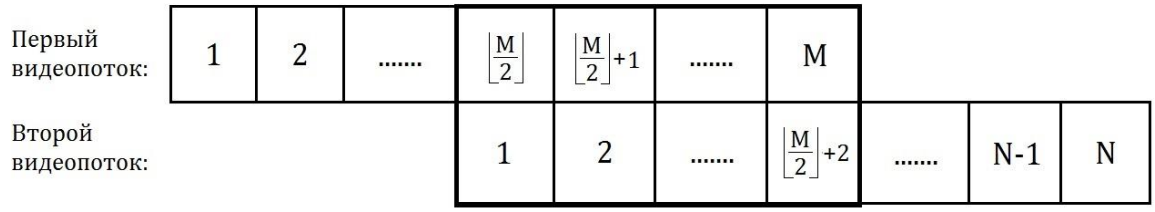

Рис. 4. Вторая итерация алгоритма

Кадры первого видеопотока смещаются до тех пор, пока не достигнут возможного минимума пересечения, но уже справа, финишная итерация выглядит следующим образом (рис. 5):

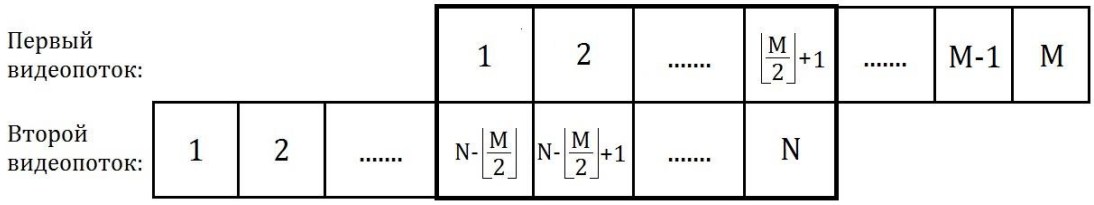

Рис. 5. Последняя итерация алгоритма

Очевидно, что подобный алгоритм гарантирует перебор всех возможных позиций видеопотоков на временной шкале, которые удовлетворяют начальным условиям.

Для каждой пары подпоследовательностей, порожденной итерацией, рассчитывается схожесть соответствующих кадров по имеющимся индексам изображений. Результатом сравнения является вектор длины L.

Остается из найденных векторов выбрать тот, который соответствует оптимальному сдвигу. В качестве критерия оптимальности может служить норма вектора. Однако напрямую сравнивать нормы векторов нельзя: они имеют разную длину, и без предварительной обработки результат будет некорректным. Поэтому сначала нужно произвести нормализацию вектора, например, поделив все элементы вектора на его длину. И уже после выбрать оптимальное

значение нормы, с помощью которого и узнать значение временного сдвига.

Алгоритм, который был описан, имеет в своей структуре две позиции произвольного выбора:

1) метод индексации и сравнения изображений;

2) норма вектора.

Чтобы найти оптимальный вариант, требуется испробовать несколько вариантов по каждой из этих позиций.

Сравнение изображений

В первой главе рассматривались методы индексации изображений. Согласно описанному алгоритму, требуется сравнивать изображения двух видеопотоков и определять уровень схожести. Значит, нужно использовать алгоритмы, рассмотренные в первой главе, так как они решают именно эту задачу.

Первым вариантом индексации изображения является метод цветовых гистограмм. Среди методов сравнения цветовых гистограмм не стоит пробовать все варианты, достаточно выбрать один, который должен показать оптимальный результат. По результатам тестов, проведенных в первой главе, метрик цветовых гистограмм наиболее оптимальный результат показывает расстояние Бхатачария.

Вторым вариантом индексации и сравнения изображений будут матрицы переходов.

Норма вектора

В обоих выбранных методах сравнения меньшему значению сравнения соответствует большая степень схожести, а каждое сравнение двух кадров должно вносить одинаковую долю в итоговый результат сравнения последовательностей, то есть норма вектора должна быть одинаково восприимчива ко всем элементам. Поэтому никаких

предпосылок для использования каких-то нестандартных норм нет, и для тестов будут использоваться нормы:

- $\|x\|_1 = \sum_i |x_i|$, норма l_1 или манхэттенское расстояние.
- $\|x\|_1 = \sqrt{\sum_i |x_i|^2}$, норма l_2 или евклидова норма.

2. 2. 2. Алгоритм сравнения кадров внутри видеопотоков

Так как постановка нашей задачи определена для статичных камер, можно заметить несколько особенностей. Во-первых, последовательные изображения в видеопотоке являются связными. Изменение цвета большей части изображения маленькое, сильное изменение цвета происходит лишь в небольших областях. Во-вторых, если в видео ничего не происходит (в поле зрения камеры не попадают какие-либо действия), то изображение практически полностью остается тем же благодаря статичности камеры.

Изучив имеющиеся исследования по данной тематике [2], [10], [11], можно прийти к выводу, что синхронизацию видеопотоков можно проводить на сравнении динамики изменяемости изображения внутри каждого видео. Это логично, ведь камеры имеют общую область захвата и если у одной камеры изображение статично, то и у другой камеры большая часть изображения будет статичной. Если же в поле зрения одной камеры попадают какие-то действия, то они попадают и в поле зрения другой камеры.

Чтобы проводить синхронизацию на основе этих данных, требуется проводить сравнение изображений внутри видеопотоков.

Алгоритм описывается следующей последовательностью шагов:

1. Проиндексировать все кадры обоих видеопотоков.
2. Внутри каждого видео сравнить попарно кадры, идущие последовательно, построить векторы, характеризующие динамику видео.

3. Сравнить подпоследовательности векторов, найти оптимальный вариант и нужный временной сдвиг.

Шаги алгоритма в математических терминах выражаются следующим образом. Пусть $M-$ количество кадров первого видеопотока, $N-$ второго. При попарном сравнении последовательных кадров для первого видео получится вектор l длины $M-1$, где l_i- результат сравнения $i-$ ого и $(i+1)-$ ого кадра. Аналогично для второго видео получится вектор p длины $N-1$.

Дальнейшая работа алгоритма идет по аналогии с алгоритмом из предыдущего параграфа. Если там проводился поиск наиболее похожих подпоследовательностей изображений, то в этот раз проводится поиск наиболее похожих подпоследовательстей скаляров из векторов l и p.

Также как и в предыдущем алгоритме, остается выбрать метод индексации и сравнения изображений и норму вектора. О вариантах, рассматриваемых автором в данной работе, рассказывается в двух следующих пунктах.

Сравнение изображений

Наиболее полезным в алгоритме может быть метод, который хорошо передает динамику изменения изображения. Среди методов, рассмотренных в первой главе, наиболее подходящим на эту роль кажется метод перцептивного хеша. Несмотря на то, что при поиске похожих изображений он показал наихудший результат, он лучше всех отражает небольшие изменения в изображениях. Последовательно идущие в видео кадры часто бывают дубликатами или очень похожими изображениями, и именно в таких классах задач перцептивный хеш показывает хорошие результаты [6].

Вторым вариантом сравнения изображений будет метод цветовых гистограмм, который работает быстрее метода матриц переходов. К

тому же, небольшие изменения на изображениях проще отследить с помощью изменения цвета, нежели текстуры.

Норма векторов

Также как и в алгоритме предыдущего параграфа, во всех трех выбранных методах сравнения меньшему значению сравнения соответствует большая степень схожести, а каждое сравнение двух кадров должно вносить одинаковую долю в итоговый результат сравнения последовательностей, то есть норма вектора должна быть одинаково восприимчива ко всем элементам. Однако здесь играет важную роль то, сколько нулевых элементов в векторе: нулевой элемент может быть более важен, потому что он указывает на полную идентичность изменений последовательных кадров двух последовательностей, и именно по этим нулевым можно эффективно сравнить изменения двух видеопотоков. Поэтому помимо тех норм, что использовались в предыдущем алгоритме, следует добавить еще одну, восприимчивую к нулевым элементам, и итоговый список норм выглядит так:

- $\|x\|_1 = \sum_i |x_i|$, норма l_1 или манхэттенское расстояние;
- $\|x\|_1 = \sqrt{\sum_i |x_i|^2}$, норма l_2 или евклидова норма;
- норма l_0 (L_0-«норма»), определяемая как количество ненулевых элементов вектора.

2. 3. Реализация и тестирование алгоритмов

В этом параграфе проводится реализация и тестирование рассмотренных ранее алгоритмов.

Для тестирования используется набор пар любительских видеозаписей, удовлетворяющих начальным условиям. Тестирование алгоритмов производилось на компьютере со следующими характеристиками:

Процессор: Intel Core i3-2350M 2.30Ghz x64

Память: 4Gb DDR3

ОС: Windows 10 x64

Код реализации алгоритмов компилировался в Visual Studio 2015 с использованием библиотеки OpenCV 2.4.9 x64 для системы Windows.

Основными характеристиками при тестировании служили время работы алгоритма синхронизации и качество синхронизации. Мерой измерения времени работы использует количество миллисекунд, которое требуется для синхронизации двух минутных видеозаписей с разрешением кадров 1280×720 пикселей и частотой 25 кадров/с. Качество синхронизации измерялось в проценте удачных синхронизаций к общему количеству испытаний. Удачной синхронизацией считалось нахождение временного сдвига, отличающегося от настоящего не более чем на 1 секунду (10):

$$|\Delta - \Delta_{\text{наст}}| < 1 \text{ сек}; \qquad (10)$$

Результаты тестирования алгоритма прямого сравнения кадров приведены в таблице 5.

Таблица 5. Результаты алгоритма прямого сравнения кадров

Метод индексации изображений и вид метрики		Время работы (мс)	Качество
Цветовые гистограммы	Метрика L1	4214	35%
	Метрика L2	4400	50%
Матрицы переходов	Метрика L1	2550	42%
	Метрика L2	2656	58%

Результаты тестирования алгоритма сравнения кадров внутри видеопотоков представлены в таблице 6.

Таблица 6. Результаты алгоритма сравнения кадров внутри видеопотоков

Метод индексации изображений и вид нормы		Время работы(мс)	Качество
Цветовые гистограммы	Норма L_1	103	60%
	Норма L_2	207	60%
	Расстояние Хэмминга	24	20%
Перцептивный хеш	Норма L_1	103	66%
	Норма L_2	207	66%
	Расстояние Хэмминга	24	33%

Анализ результатов тестирования

Первый алгоритм показал результаты в среднем хуже второго. Метод цветовых гистограмм и метод матриц переходов показали примерно равные результаты в плане качества, в плане скорости лучше показал себя метод матриц переходов. Однако не стоит забывать, что индексация у этого метода намного дольше, чем у метода цветовых гистограмм, поэтому в суммарном времени они проигрывает.

Алгоритм сравнения кадров внутри видеопотоков по качеству показал наилучшие результаты, а если быть точнее, то вариант его реализации с использованием хешей. Метод цветовых гистограмм отстал ненамного. Так как алгоритм непосредственно самой синхронизации не зависел от метода индексации и сравнения изображений, то по времени оба варианта показали одинаковый результат. Но стоит учитывать, что сама индексация быстрее у алгоритма перцептивного хеша, поэтому также выигрывает и в скорости. Среди норм по качеству худший результат показало расстояние Хэмминга, и даже его хорошие результаты по времени не нивелируют столь существенный разрыв. Никаких различий в плане

качества нормы L_1 и L_2 не показали, но по скорости норма L_1 немного впереди.

Таким образом, наилучший результат показал алгоритм сравнения кадров внутри видеопотоков с использованием перцептивного хеша и нормы L_1.

Заключение

В представленной работе автором было проведено изучение методов индексации изображений и рассмотрено их применение для задач поиска схожих изображений и синхронизации. Благодаря проведенным тестированиям удалось сравнить между собой методы индексирования изображений путем определения скорости работы и качества результата, а также оценить алгоритмы синхронизации видеопотоков. К основным выводам относится результат тестирования алгоритмов индексирования изображений, продемонстрировавший заметное преимущество в скорости метода цветовых гистограмм перед методом матриц переходов при одинаковом качестве сравнения, а также оптимальность по качеству и времени алгоритма сравнения кадров внутри видепотоков с использованием метода перцептивных хешей.

Работа может быть продолжена путем усовершенствования и применения других модификаций к рассмотренным методам индексирования. Для задачи синхронизации могут быть ослаблены начальные условия, что позволит использовать представленные алгоритмы синхронизации для более широкого класса видеопотоков, например, имеющих разную частоту кадров или меньшую область пересечения.

Список литературы и источников

1. Гонсалес Р.С., Вудс Р. Цифровая обработка изображений. Издание 3-е, исправленное и дополненное. Москва: Техносфера, 2012. 1104 с.
2. Понс Ж., Форсайт Н.А. Компьютерное зрение. Современный подход. М.: Издательский дом «Вильямс», 2004. 928 с.
3. Новейшие методы обработки изображений / Под ред. А.А.Потапова. М.: ФИЗМАТЛИТ, 2008. 496 с.
4. Ярышев С.Н. Цифровые методы обработки видеоинформации и видеоаналитика: учебное пособие. СПб.: СПбГУ ИТМО, 2011. 83 с.
5. A. Murat Tekalp. Digital Video Processing, Second Edition. Publisher: Prentice Hall, 2015. 595 p.
6. Начало работы с библиотекой OpenCV. http://www.intuit.ru/studies/courses/10621/1105/lecture/17985
7. OpenCV шаг за шагом. http://robocraft.ru/page/opencv/
8. Денисюк В.С. Алгоритмы выделения особенностей на изображениях с целью классификации заболеваний растений // Конструирование и оптимизация параллельных программ / Под ред. В.Н.Касьянова. Новосибирск: Институт систем информатики им. Ершова, 2008. С.71-82.
9. Байгарова Н.С., Бухштаб Ю.А., Горный А.А., Евтеева Н.Н., Лялин В.Ю., Монастырский А.В., Стрелков А.Ю. Методы индексирования и поиска изображений и видеоданных на основании визуального содержания // Институт прикладной математики им. М.В.Келдыша РАН, 2002. С.263-273.
10. Gregg Pass, Ramin Zabih. Comparing Images Using Joint Histograms // Regular paper**,** Multimedia Systems, May 1999, Volume 7, Issue 3, pp 234-240.
11. Поиск подобных изображений в базе данных индексированных видео. http://web.snauka.ru/issues/2014/05/34690

12. Построение признаков и сравнение изображений: глобальные признаки. https://habrahabr.ru/company/yandex/blog/255627
13. Рудаков И.В., Васютович И.М. Исследование перцептивных хеш-функций изображений // Наука и образование. МГТУ им. Н.Э.Баумана. Электронный журнал 2015. №8. С. 269-280.
14. Выглядит похоже. Как работает перцептивный хеш. https://habrahabr.ru/post/120562
15. Как бороться с репостами или пара слов о перцептивных хешах. https://habrahabr.ru/post/237307
16. OpenCV 2.4.9 Docementation. http://docs.opencv.org/2.4.9
17. Котов А.С. Индексирование видеопотока на основе выделения лиц // Пояснительная записка к ВКР, ТУСУР, 2008. 109 с.
18. Irena Koprinska, Sergio Carrato. Temporal video segmentation: A survey// Signal Processing: Image Communication, 16, 2001, P. 477 - 500